COEN 280 Database Systems Fall 2022

Project (200 pts) + 10 pts EC

Note: Requirements given below from 1.0 to 3.0 are for **Single person teams**.

Two (more than 2 people per team **not** accepted) people teams should, in addition to requirements from 1.0 to 3.0, implement requirements in 3.1.

1.0 Introduction

You are commissioned to design and implement a **Sales Management System** for ***MasterRobot***, **a bookstore** that sells children's comic books and Cartoon movies online. The Sales Management System uses a Relational Database to maintain the sales and customer records and support queries to track customer orders.

You are required to implement the system using an Oracle database management system.

The store sells **comic books** and **cartoon movies** (based on the comic books**)** online. Comic books and cartoon movies are considered **StoreItems**. A comic book or a cartoon may have 0 or more copies.

All **StoreItems** have an *ItemId* **(unique)**, *price* and *no.of copies***.** Each **comic book** has a *title* and *publishedDate*. A **cartoon movie** has a *title*, *studio name* and *description*.

A **customer** can be a **regular customer** (*Custid, custType,name, phone/email, address*) or a **gold customer**. A **gold customer**, in addition to having all the attributes of a customer, also has additional attributes, *dateJoined* and *coupons*. A gold customer pays an *annual fee* and does not have to pay the shipping fee on her/his orders.

When a customer (or a gold customer) orders StoreItems (books or movies), a **CustOrder** is created. A **CustOrder (Note:** *order by* is a SQL key word**)** contains the *orderid* (unique) and is **associated** with a number of **OrderLineItem**s, *date of order*, *shippedDate* and *shippingFee*. An **OrderLineItem** has a *line id* (unique for that order, for example, lineid 1,2,3 etc). Each **OrderLineItem** is associated with a **StoreItem** and *quantity*.

Each customer's order includes a flat rate of 5% tax (for all customers) on the order total.

GoldCustomers have no shipping fee and 10% discount on an order of $100 or more.

1.1 Constraints to be enforced:

a) The custType of a customer can be of only one of the two values, namely, '**regular**', or **'gold**'.
b) Phone (or email) must be unique and not null.
c) The no.of copies of a comicBook or a cartoon movie cannot be < 0.
d) The no. of copies of any book (or movie) ordered cannot be more than the available no. of copies of that item.
e) The shippedDate cannot be less than the OrderedDate.
f) When a regular customer orders books (and/or movies), the shipping fee is $10. Before the items are shipped (ie. the shippedDate is null), if the custType of that customer changes to 'gold', then the shipping fee must be changed to 0 on all her/his orders that are not shipped yet.

2.0 Functionality and Queries to be implemented

a) Create a few customers (a mix of regular and gold customers), a few comic Books, and cartoons (with data of your choice). A minimum of 10 tuples for each are expected.

b) Write a **PLSQL function** (for example, **createCustOrde**r())to create a **CustOrder**, where the *orderId* is randomly generated by the function and return the *orderId* to the calling program.

c) Write a **PLSQL procedure**, let us call it, **createOrderLineItem()** that takes several parameters – *custOrderId, itemid, customerid, date ordered, number ordered and shipped date*). **Note**: You are free to add any other parameters).

The procedure must do the following:

- Check if the no.of items (books or movies) ordered is <= no. of copies available for that item. If not, take an appropriate action (for example, display an error message and exit).
- Check if the customer is regular or gold member. If gold member, make shipping fee 0. Otherwise, add a flat shipping fee of $10.00.
- Create an OrderLineItem. Note: Make the shippedDate NULL (this can be changed later). Note: Each OrderLineItem should have a reference to the CustOrder it belongs to.
- Update the no. of copies in the item table.

d) Write (and test) a **Trigger** that does the following:

   After the custType in Customer is updated to 'gold', then check if that customer has any orders pending (not shipped yet) and set the *shippingFee* to 0.

e) Write a PLSQL procedure, **setShippingDate()**, that given an *orderid* and *shipping date* as parameters, sets the *shippingDate* for that order.

f) Write a PLSQL function, **computeTotal()** that takes an *orderid*, computes the total for that order and returns the total. The total is the grand total price of all the OrderLineItems of this custorder. The function should consider the customer type, tax, shipping fee etc, to compute the total.

g) Write a PLSQL procedure **showItemOrders()** that takes a *custOrderId* as a parameter and displays the **details of that CustOrder**.

   The details of a **CustOrder** should include the details of **Customer**, details of Items **ordered**, and **payment** details. **Customer details** include custid, name, phone and address.
   For each CustOrder, show Orderid, details of each Line item in the order, date Ordered and shipped date. **Payment details** include the total for all order line items, tax, shipping fee, any discount applied (in case of Gold Customers) and the grand total.

Extra Credit (10 pts)

h) Write a PLSQL procedure **showItemOrdersAfter()** that takes a *customerid* and a *date* as parameters and displays the **details of each CustOrder after** the date given as a parameter.

The details of each **CustOrder** should be the same as in g).